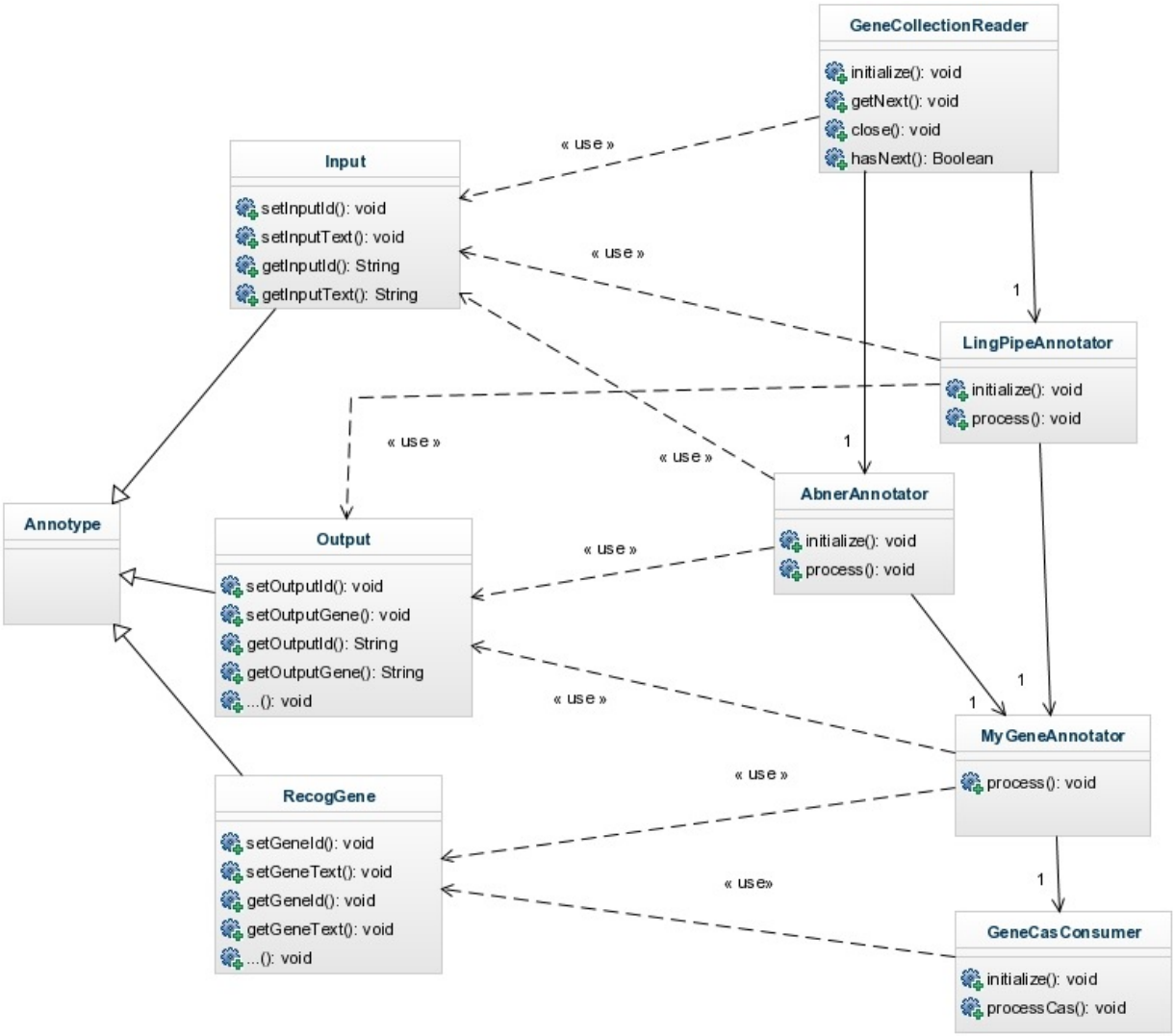
Hongfei Zhang
Andrew ID: hongfeiz
October 10, 2014

HW2 Report

1. Architecture Design

1.1 Design Class Diagram



1.2 Type System

The type system descriptor of my project is located in the file ./src/main/resources/descriptors/deiis_types.xml. There is a super-type and three child-types.

1.2.1 edu.cmu.deiis.types.Annotype

It is defined with two features casProcessorId <uima.cas.String> and confidence <uima.cas.double>, which identify the analysis engines used in aggregation and describe their confidences.

1.2.2 edu.cmu.deiis.types.Input

Inherited from edu.cmu.deiis.types.Annotype, I add two features into it: inputId <uima.cas.String> and inputText <uima.cas.String>, which described the ID and content of an Input type that generated by collection reader and put into analysis engines.

1.2.3 edu.cmu.deiis.types.Output

Inherited from edu.cmu.deiis.types.Annotype, I add two features into it: outputId <uima.cas.String> and outputGene <uima.cas.String>, which described the ID and content of an Output type that generated by AbnerAnnotator analysis engines and LingPipeAnnotator analysis engines, and put into a MyGeneAnnotator analysis engine.

1.2.4 edu.cmu.deiis.types.RecogGene

Inherited from edu.cmu.deiis.types.Annotype, I add two features into it: geneId <uima.cas.String> and geneText <uima.cas.String>, which described the ID and content of a RecogGene type that generated by MyGeneAnnotator analysis engine.

1.3 Descriptors

1.3.1 Collection Reader Descriptor

File path: ./src/main/resources/descriptors/collectionReaderDescriptor.xml

It sets parameters of collection reader which is hw2.in as InputFile in my project. It uses type system defined in deiis_types.xml and take the type Input as output.

1.3.2 Analysis Engine 1 (abnerDescriptor)

File path: ./src/main/resources/descriptors/abnerDescriptor.xml

It uses type system defined in deiis_types.xml and take the type Input as input and type Output as output.

1.3.3 Analysis Engine 2 (lingPipeDescriptor)

File path: ./src/main/resources/descriptors/lingPipeDescriptor.xml

It also uses type system defined in deiis_types.xml and take the type Input as input and type Output as output.

1.3.4 Analysis Engine 3 (myDescriptor)

File path: ./src/main/resources/descriptors/myDescriptor.xml

It uses type system defined in deiis_types.xml and take the type Output as input and type RecogGene as output.

1.3.5 Aggregate Analysis Engine

File path: ./src/main/resources/descriptors/aaeDescriptor.xml

It aggregates the three AEs above, and fixes their order as:

1) abnerDescriptor
2) lingPipeDescriptor
3) myDescriptor

1.3.6 Cas Consumer Descriptor

File path: ./src/main/resources/descriptors/casConsumerDescriptor.xml

It sets parameters of Cas consumer which is hw2-hongfeiz.out as OutputFile in my project. It uses type system defined in deiis_types.xml and take the type RecogGene as input.

1.4 PipeLine Design

1.4.1 Collection Reader

File path: ./src/main/java/collectionreader/GeneCollectionReader.java

It extends from CollectionReader_ImplBase.

In the initialize() method, it calls parameter “InputFile” defined in collection reader descriptor. It reads document from input file “hw2.in” line by line and splits each line into two parts: id and text. It call getNext() method to put id and text into Input type of Cas. Methods getProgress() and hasNext() are also overridden.

1.4.2 Abner Annotator

File path: ./src/main/java/annotator/AbnerAnnotator.java

It extends from JCasAnnotator_ImplBase.

It calls initialize() to initialize a Tagger object tagger used in process(). It calls process() method to recognize gene mention tags in each sentence.

1.4.3 LingPipe Annotator

File path: ./src/main/java/annotator/LingPipeAnnotator.java

It extends from JCasAnnotator_ImplBase.

In initialize(), it initializes a ConfidenceChunker object chunker used in process() and loads dictionary file used for LingPipe NER. It calls process() method to recognize gene mention tags in each sentence.

1.4.4 MyGene Annotator

File path: ./src/main/java/annotator/MyGeneAnnotator.java

It extends from JCasAnnotator_ImplBase.

It calls process() method to determine which gene mention tags processed by Abner and LingPipe should be exported in RecogGene type and delivered to Cas consumer.

1.4.5 Cas Consumer

File path: ./src/main/java/casconsumer/GeneCasConsumer.java
it extends from CasConsumer_ImplBase.
In the initialize() method, it calls parameter “OutputFile” defined in Cas consumer descriptor. In process() method, it writes recognized gene mention tags into the destination file “hw2-hongfeiz.out”

2. Algorithm Design

2.1 Abner NER

In the process() method, it applies FSIterator to get Input type from JCas. It calls Tagger.getEntities() method to process each sentence and split them into several named entities. Regular expression is used, as [0-9a-zA-z-\\s]+, to exclude tags with other strange characters. Offset is also handled in the method. The result is exported in Output type.

2.2 LingPipe NER

In the initialize() method, it use resource loader to load dictionary file “selftrain.dic”, which is trained by LingPipe using its API. In the process() method, it applies FSIterator to get Input type from JCas. ConfidenceChunker.nBestChunks() method is used to generate both recognized text and its confidence. Offset is also handled in the method. The result is exported in Output type.

2.3 My Gene Annotator

In the process() method, it applies FSIterator to get Output type from JCas. The tags predominantly come from LingPipe. If the confidence of LingPipe result is larger than a value, just output it. If the confidence is within a lower interval, it compares results from both Abner and LingPipe, and output the tags which exist in both NER tool results. I use a hash map to implement the algorithm. It firstly stores results of Abner NER into a hash map begin2end, taking the <begin, end> as <key, value>, and then compare the <begin, end> pairs of LingPipe results with the elements in

begin2end. If they are identical, the tags should be output. The processing result of the annotator is exported in RecogGene type.

3. Performance Evaluation

3.1 Taking the confidence intervals of LingPipe as (0.5, 0.99] and (0.99,1], N = 40

It means if confidence>0.99, just output; else if 0.5<confidence<=0.99, and Abner has the same result, output; else don't output. And the parameter N in ConfidenceChunker.nBestChunks() method is set as 40.

The running outcome of grader.sh is as follows:

Precision: 0.882308845577

Recall: 0.644401861484

F1 Score: 0.744818857776

3.2 Taking the confidence intervals of LingPipe as (0.35, 0.99] and (0.99,1], N = 40

The running outcome of grader.sh is as follows:

Precision: 0.864361510142

Recall: 0.655570763756

F1 Score: 0.745625505947

3.3 Taking the confidence intervals of LingPipe as (0.35, 0.8] and (0.8,1], N = 40

The running outcome of grader.sh is as follows:

Precision: 0.823170731707

Recall: 0.790856830003

F1 Score: 0.806690307989

3.4 Taking the confidence intervals of LingPipe as (0.35, 0.7] and (0.7,1], N = 40

The running outcome of grader.sh is as follows:

Precision: 0.803575297941

Recall: 0.812154393649

F1 Score: 0.807842069435

3.5 Taking the confidence intervals of LingPipe as (0.35, 0.7] and (0.7,1], N = 20

The running outcome of grader.sh is as follows:

Precision: 0.803459119497

Recall: 0.811333150835

F1 Score: 0.807376937536

3.6 Taking the confidence intervals of LingPipe as (0.35, 0.7] and (0.7,1], N = 10

The running outcome of grader.sh is as follows:

Precision: 0.803399248243

Recall: 0.807445934848

F1 Score: 0.805417508601

Consider all these outcomes: the larger N comes the larger performance, and the bigger confidence upper bound of solely using LingPipe comes the more precision yet less recall.

As a result, my final submission set confidence intervals of LingPipe as (0.35, 0.7] and (0.7,1], N = 40